

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**

**VIỆN KỸ THUẬT**

**BÁO CÁO CUỐI MÔN HỌC**

**MÔN HỌC CÁC HỆ THỐNG MÁY TÍNH ĐỀU KHIỂN**

**Ngành: Robot và Trí Tuệ Nhân Tạo**

**Sinh viên: Nguyễn Đình Khiêm – 2186300559**

**Lớp: 21DRTA1**

*TP.HCM, Ngày 2, Tháng 6, Năm 2023*

MỤC LỤC

[I. TỔNG QUAN BÀI BÁO CÁO: 4](#_Toc138273871)

[II. MỤC TIÊU – PHƯƠNG PHÁP: 4](#_Toc138273872)

[III. KẾT QUẢ - THẢO LUẬN: 4](#_Toc138273873)

[1. Báo cáo cá nhân: 4](#_Toc138273874)

[1.1. Ưu – nhược điểm của Hệ điều hành Windows và Linux: 4](#_Toc138273875)

[1.1.1. Windows: 4](#_Toc138273876)

[1.1.2. Linux: 4](#_Toc138273877)

[1.2. Các hệ điều hành dùng trên Robot: 5](#_Toc138273878)

[1.3. Các bước cài đặt hệ điều hành trên Raspberry PI: 5](#_Toc138273879)

[1.3.1. Chuẩn bị: 5](#_Toc138273880)

[1.3.2. Tải hệ điều hành: 5](#_Toc138273881)

[1.3.3. Ghi ảnh vào thẻ nhớ: 5](#_Toc138273882)

[1.3.4. Kết nối và cài đặt: 6](#_Toc138273883)

[1.3.5. Cấu hình ban đầu: 6](#_Toc138273884)

[1.4. Các câu lệnh cài thư viện cơ bản: 6](#_Toc138273885)

[1.5. Ba ví dụ về xử lý ảnh: 6](#_Toc138273886)

[1.5.1. Tìm tọa độ của vật thể: 6](#_Toc138273887)

[1.5.2. Xác định vật thể có màu đỏ: 8](#_Toc138273888)

[1.5.3. Xác định hình dạng vật thể: 9](#_Toc138273889)

[2. Báo cáo nhóm: 10](#_Toc138273890)

[2.1.1. Bài 1 – Giao tiếp máy tính nhúng Rasberry PI với vi điều khiển Arduino: 10](#_Toc138273891)

[2.1.2. Nguyên lý hoạt động: 10](#_Toc138273892)

[2.1.3. Chương trình phần mềm: 11](#_Toc138273893)

[2.1.4. Kết quả: 13](#_Toc138273894)

[2.2. Bài 2 – Thiết kế giao diện giao tiếp với vi điều khiển Arduino bằng ngôn ngữ Python: 13](#_Toc138273895)

[2.2.1. Nguyên lý hoạt động: 14](#_Toc138273896)

[2.2.2. Chương trình phần mềm: 14](#_Toc138273897)

[2.2.3. Kết quả: 16](#_Toc138273898)

[2.3. Đồ án cuối môn học –Hệ thống mở cửa bằng vân tay: 16](#_Toc138273899)

[2.3.1. Ý tưởng – Mục tiêu: 16](#_Toc138273900)

[2.3.2. Nguyên lý hoạt động: 16](#_Toc138273901)

[2.3.3. Sơ đồ kết nối phần cứng: 16](#_Toc138273902)

[2.3.4. Chương trình phần mềm: 16](#_Toc138273903)

[2.3.5. Kết quả - Thảo luận - Hướng phát triển: 16](#_Toc138273904)

[IV. TÀI LIỆU THAM KHẢO: 16](#_Toc138273905)

# TỔNG QUAN BÀI BÁO CÁO:

Đây là bài báo cáo cuối môn học của nhóm 7,tổng hợp lại những kiến thức đã học trong môn học này. Bài báo cáo cuối môn học nhằm đánh giá sự hiểu biết, khả năng nghiên cứu và khả năng áp dụng kiến thức đã học trong môn học. Bài báo cáo gồm 3 phần chính, với 2 bài tập đã làm được từ trên trường và 1 đồ án cuối môn học.

# MỤC TIÊU – PHƯƠNG PHÁP:

# KẾT QUẢ - THẢO LUẬN:

## Báo cáo cá nhân:

### Ưu – nhược điểm của Hệ điều hành Windows và Linux:

#### Windows:

* Ưu điểm:
  + **Thân thiện với người dùng:** Windows có giao diện đồ họa dễ sử dụng, phổ biến và quen thuộc với nhiều người dùng.
  + **Tích hợp nhiều tiện ích:** Windows có sẵn một loạt phần mềm ứng dụng phong phú và đa dạng, bao gồm cả phần mềm thương mại và miễn phí.
  + **Tương thích phần cứng:** Windows có hỗ trợ rộng rãi cho nhiều loại phần cứng và thiết bị ngoại vi khác nhau, đảm bảo tính tương thích cao với các thiết bị mới nhất trên thị trường.
* Nhược điểm:
  + **Giá cả:** Windows là một phần mềm thương mại, nghĩa là người dùng phải mua bản quyền để sử dụng. Điều này có thể là một hạn chế cho những người không muốn bỏ tiền để có một hệ điều hành.
  + **Bảo mật:** Windows đã gặp phải nhiều vấn đề liên quan đến bảo mật và trở thành mục tiêu của các phần mềm độc hại và các cuộc tấn công mạng.
  + **Quyền kiểm soát:** Windows có thiên hướng giới hạn quyền kiểm soát của người dùng, điều này có thể khiến người dùng cảm thấy hạn chế trong việc tùy chỉnh và điều chỉnh hệ thống.
  + **Dung lượng lớn:** Những file của Windows bao gồm cả cập nhật và để chạy thường có dung lượng khá nặng, không thích hợp với những máy tích có ít dung lượng.

#### Linux:

* Ưu điểm:
  + **Mã nguồn mở:** Linux là một hệ điều hành miễn phí và nguồn mở, cho phép người dùng sửa đổi, tùy chỉnh và phân phối lại theo nhu cầu của họ.
  + **Bảo mật:** Linux có một cộng đồng lớn và tích cực, giúp phát hiện và khắc phục các lỗ hổng bảo mật nhanh chóng. Nó cũng cho phép người dùng có quyền kiểm soát cao hơn đối với quyền truy cập và phân quyền.
  + **Ổn định và hiệu suất**: Linux được coi là ổn định và tin cậy, thường có khả năng hoạt động lâu dài mà không cần khởi động lại. Nó cũng tối ưu hóa hiệu suất cho máy tính.
  + **Dung lượng ít:** Linux được biết đến là hệ điều hành nhỏ gọn và sử dụng tài nguyên máy tính hiệu quả. Nó phù hợp cho các máy tính nhúng mini.
* Nhược điểm:
  + **Khó sử dụng đối với người mới:** Linux thường khá xa lạ và khó dùng đối với người mới, đặc biệt là những người không quen với dòng lệnh và không có kinh nghiệm với hệ thống mã nguồn mở.
  + **Hạn chế tuơng thích phần cứng**: Mặc dù đã có sự tiến bộ đáng kể trong việc hỗ trợ phần cứng, nhưng vẫn có một số phần cứng không tương thích hoặc không có driver tương ứng cho Linux.
  + **Sự đa dạng và không đồng nhất:** Vì Linux là mã nguồn mở và có nhiều phiên bản phân phối khác nhau, điều này có thể dẫn đến sự đa dạng và không đồng nhất trong cách cài đặt, cấu hình và sử dụng các phiên bản Linux khác nhau.

### Các hệ điều hành dùng trên Robot:

Có một số hệ điều hành được sử dụng phổ biến trên robot. Tiêu biểu phải kể đến:

* **Robot Operating System (ROS):** ROS là một hệ điều hành mã nguồn mở và được sử dụng rộng rãi trong ngành robot. Nó cung cấp một cơ sở hạ tầng mạnh mẽ để phát triển, kiểm tra và điều khiển các ứng dụng robot.
* **Ubuntu:** Ubuntu là một hệ điều hành Linux phổ biến và được sử dụng trên nhiều loại robot. Ubuntu cung cấp sự ổn định, bảo mật và hỗ trợ phần mềm đáng tin cậy.
* **Windows IoT Core:** Windows IoT Core là một phiên bản của hệ điều hành Windows được tối ưu cho các thiết bị nhúng và robot. Nó cung cấp tính tương thích với nền tảng Windows và các công cụ phát triển quen thuộc của Microsoft.
* **Android:** Android mặc dù chủ yếu được sử dụng trên điện thoại di động và máy tính bảng, nhưng cũng được sử dụng trong một số robot di động. Android cung cấp một môi trường phát triển ứng dụng mạnh mẽ và có sẵn nhiều tài nguyên và thư viện hỗ trợ.

### Các bước cài đặt hệ điều hành trên Raspberry PI:

#### Chuẩn bị:

* Raspberry Pi: Một Raspberry Pi và nguồn cấp điện phù hợp.
* Thẻ nhớ microSD: Hệ điều hành sẽ được cài đặt và chạy từ thẻ nhớ microSD. Thẻ nhớ phải có dung lượng đủ và được format đúng.
* Máy tính: Một máy tính để tải xuống và chuẩn bị hệ điều hành.

#### Tải hệ điều hành:

* Truy cập vào trang web chính thức của Raspberry Pi hoặc trang web của hệ điều hành bạn muốn cài đặt.
* Tìm và tải xuống hình ảnh (image) của hệ điều hành phù hợp với Raspberry Pi. Các hệ điều hành phổ biến cho Raspberry Pi bao gồm Raspbian (hoặc Raspberry Pi OS), Ubuntu Mate, và các phiên bản Linux khác.

#### Ghi ảnh vào thẻ nhớ:

* Sử dụng phần mềm ghi hình Raspberry Pi Imager để ghi hình ảnh vào thẻ nhớ microSD.
* Chọn hình ảnh (image) đã tải xuống và chọn thẻ nhớ microSD làm thiết bị đích.
* Tiến hành ghi hình ảnh và đợi quá trình hoàn tất.

#### Kết nối và cài đặt:

* Kết nối thẻ nhớ microSD đã chứa hình ảnh cài đặt vào Raspberry Pi.
* Kết nối các thiết bị ngoại vi cần thiết như bàn phím, chuột, màn hình (hoặc kết nối qua SSH từ máy tính).
* Kết nối nguồn cấp điện vào Raspberry Pi để bắt đầu quá trình cài đặt.
* Raspberry Pi sẽ khởi động và tiến hành cài đặt hệ điều hành từ thẻ nhớ microSD. Quá trình này có thể mất một thời gian.

#### Cấu hình ban đầu:

Sau khi quá trình cài đặt hoàn tất, bạn sẽ được yêu cầu cấu hình ban đầu của hệ điều hành. Theo hướng dẫn trên màn hình để cấu hình ngôn ngữ, mạng, tài khoản người dùng và các thiết lập khác.

### Các câu lệnh cài thư viện cơ bản:

* Sử dụng apt (Advanced Package Tool) trên các hệ điều hành dựa trên Debian, như Ubuntu: ***sudo apt install tên\_thư\_viện***

*Ví dụ: “sudo apt install zlib1g-dev” để cài đặt thư viện zlib.*

* Sử dụng pip (Package Installer for Python) để cài đặt các thư viện Python: ***pip install tên\_thư\_viện***

*Ví dụ: “pip install requests” để cài đặt thư viện requests trong Python.*

* Sử dụng apt-get trên các phiên bản cũ hơn của Ubuntu: ***sudo apt-get install tên\_thư\_viện***

*Ví dụ: “sudo apt-get install zlib1g-dev” để cài đặt thư viện zlib trên Ubuntu.*

* Sử dụng snap: Một số ứng dụng có thể được cài đặt thông qua snap, một hình thức cài đặt ứng dụng độc lập trên Linux. Sử dụng lệnh sau để cài đặt một ứng dụng từ snap: ***sudo snap install tên\_ứng\_dụng***

*Ví dụ: “sudo snap install vscode” để cài đặt Visual Studio Code thông qua snap.*

### Ba ví dụ về xử lý ảnh:

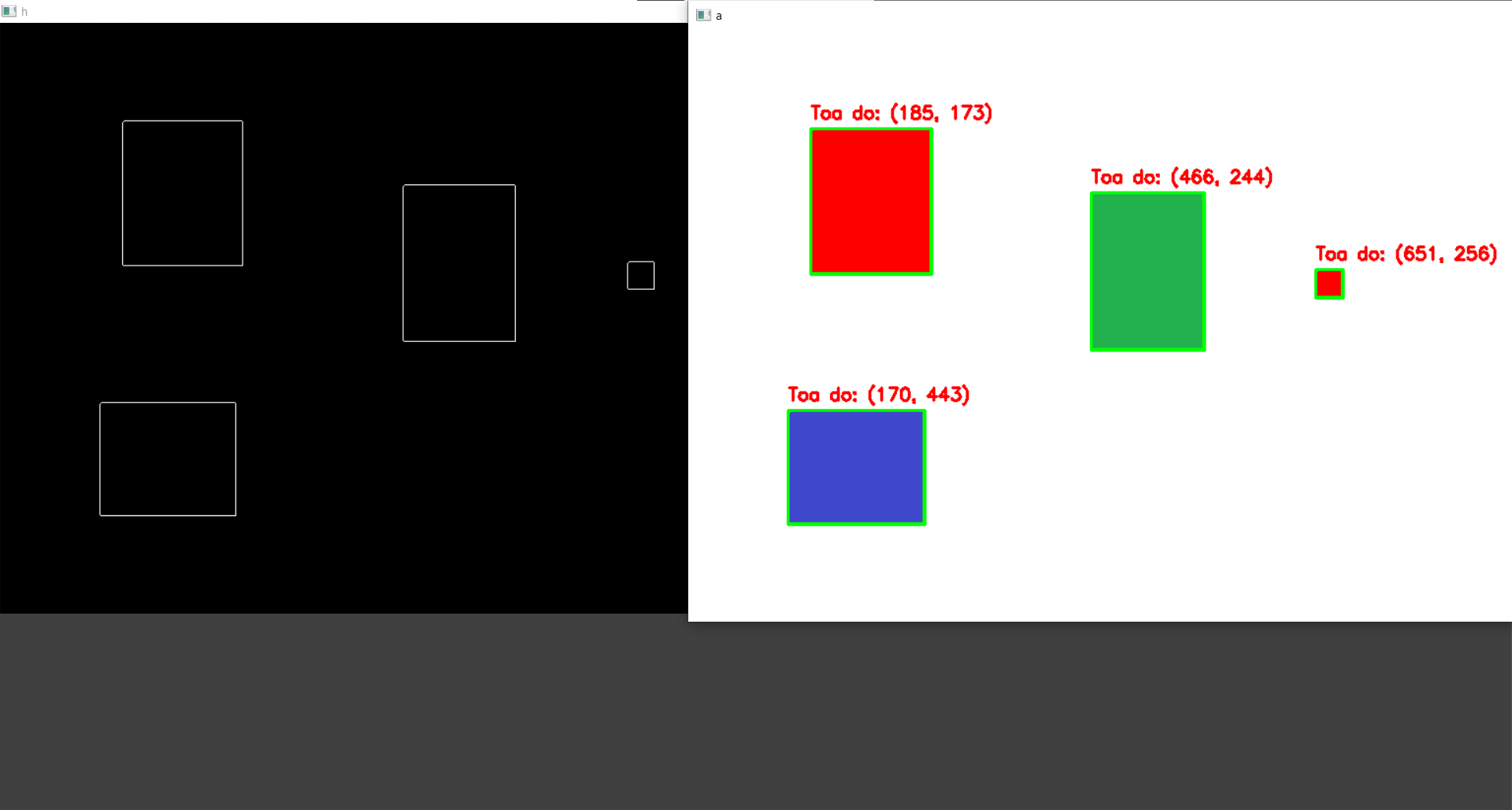
Tất cả những xưa lý ảnh này đều được thực hiện bằng thư viện OpenCV được viết bằng Python

#### Tìm tọa độ của vật thể:

Để tìm tọa độ (pixel) của 1 vật thể trong ảnh, chúng tôi dùng thuật toán Canny. Thuật toán Canny là một thuật toán phát hiện cạnh trong xử lý ảnh. Nó được đặt tên theo tên của nhà nghiên cứu John F. Canny và đã trở thành một trong những phương pháp phổ biến và mạnh mẽ để tìm cạnh trong ảnh.



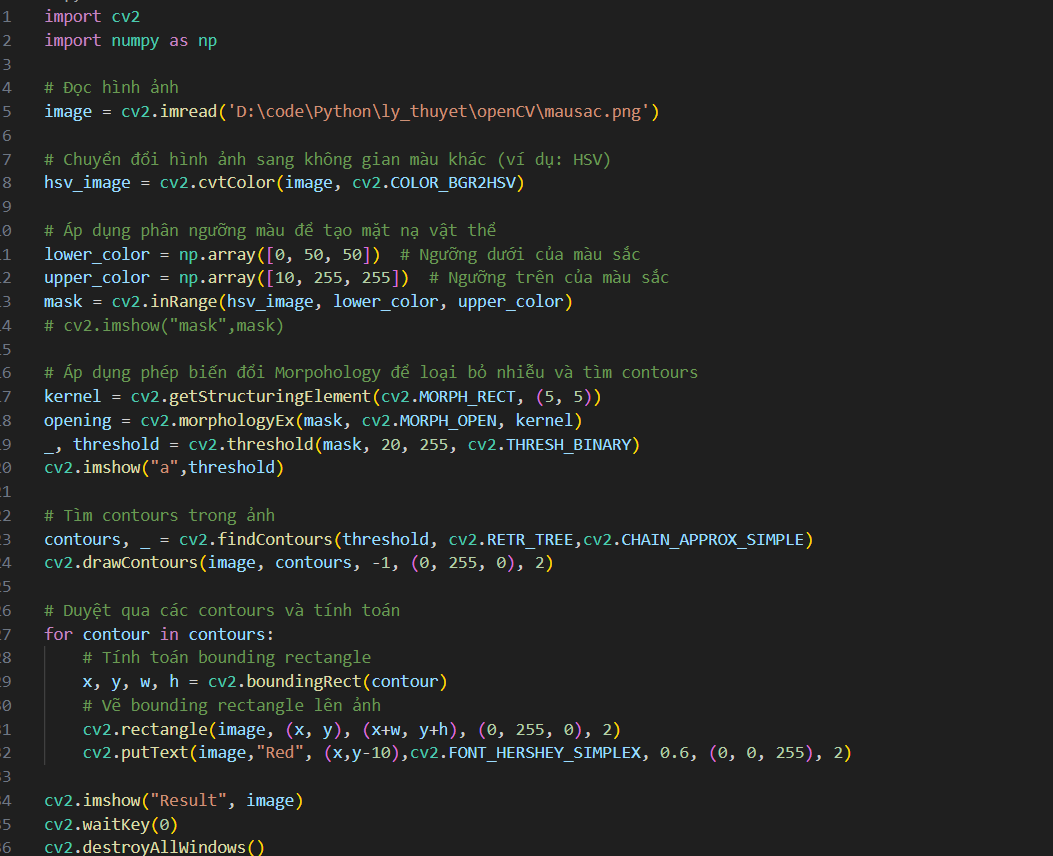
* Chúng tôi đọc hình ảnh cần xử lý sau đó chuyển nó thành ảnh xám. Sau đó thực hiện xử lý ảnh rồi lưu lại vào biến “edges”. Biến này sẽ tìm tất cả các cạnh của những vật thể có trong ảnh.
* Sau khi có những cạnh của vật thể, tôi tiến thành vẽ những đường viền xung quanh vật thể. Biến “contours” lưu lại những đường viền đó.
* Biến “contours” chứa tạ hợp tất cả các contour của các vật thể. Tôi duyệt qua từng contour để tính toán. Bằng cách lấy tọa độ y,x (tọa độ của vật thể) cộng cho chiều dài, rộng chia 2 ta được tâm của vật thể.
* In tọa độ của vật thể lên ảnh gốc bằng hàm putText.



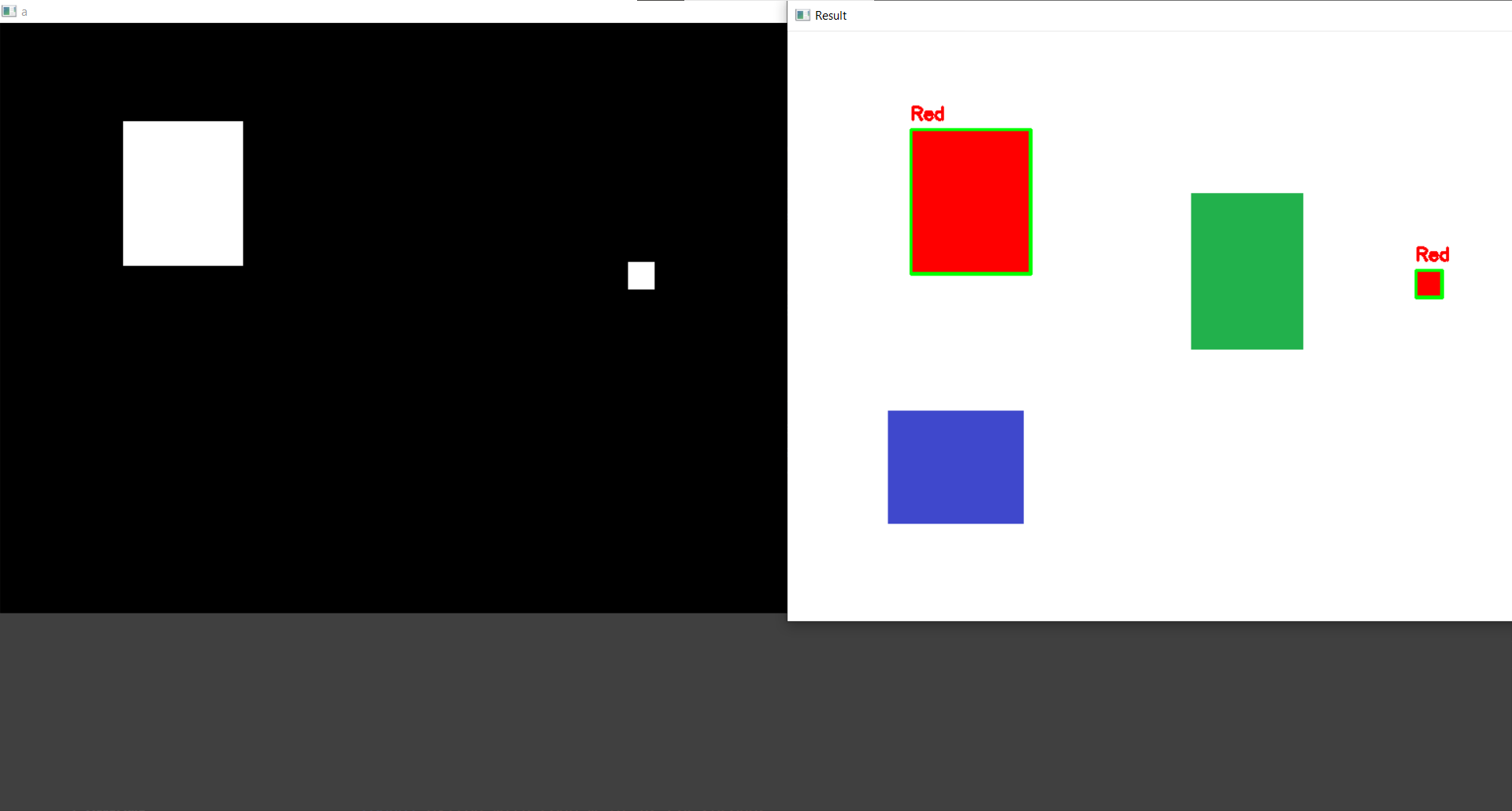
Hình bên trái là mặt nạ (mask) được thuật toán Canny nhận diện vật thể. Hình bên phải là kết quả.

#### Xác định vật thể có màu đỏ:

Để xác định vật thể có màu đỏ, chúng tôi dùng phương pháp phân ngưỡng màu.Tức là những vật thể nào có giá trị nằm trong khoảng màu mà tôi cấu hình thì đều được nhận diện.



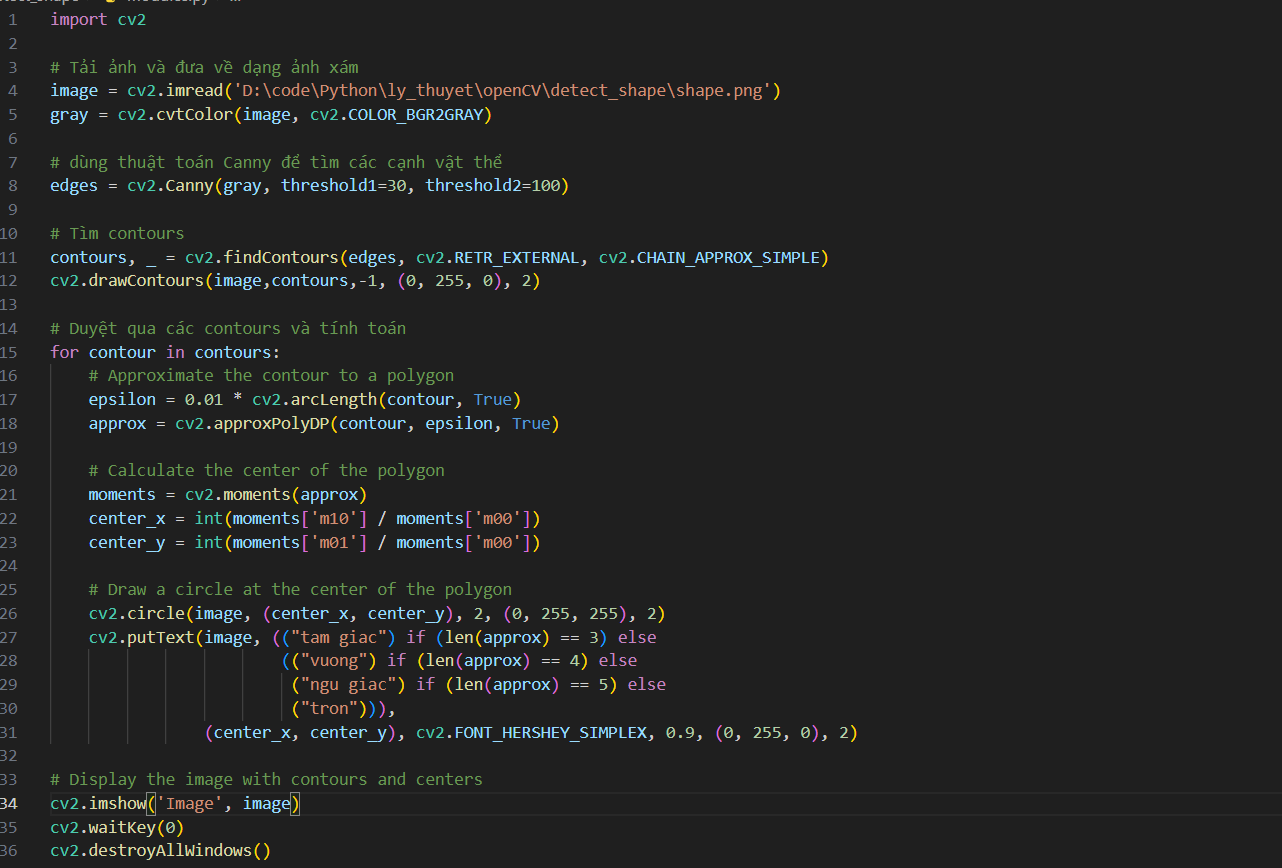
* Tôi chuyển hình ảnh sang dạng HSV để xử lý. Đây là hệ màu được dùng phổ biến trong xử lý ảnh. Sau đó tôi tạo mặt nạ bằng phương pháp **phân ngưỡng màu** bằng hàm ***cv2.inRange().***
* Để đem lại độ hiệu quả cho việc xử lý ảnh, tôi thêm phép biến đổi Morpohology, đây là một phương pháp quan trọng trong xử lý ảnh để cải thiện và thay đổi hình dạng của các vùng trong ảnh dựa trên cấu trúc và kích thước của các phần tử gọi là phần tử cấu trúc (structuring element). Nó được dùng để loại bỏ nhiễu giúp cho việc nhận diện vật thể chính xác hơn.
* Sau khi có được contours, tôi vẽ nó lên ảnh. Với mỗi contour, tôi đóng khung và in lên ảnh màu đỏ (red)



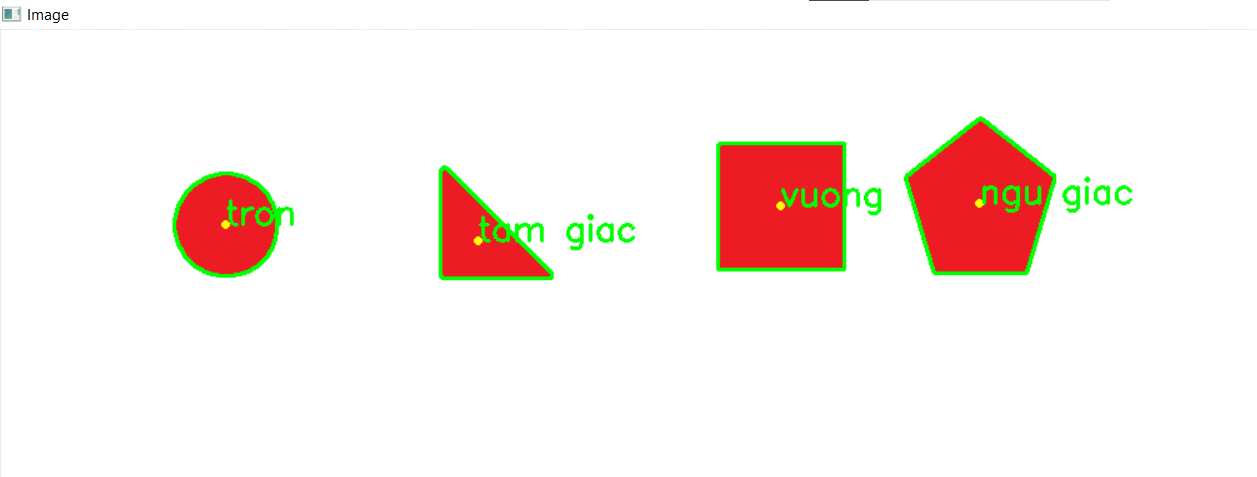
Hình bên trái là mặt nạ (mask) được phương pháp phân ngưỡng màu nhận diện vật thể. Hình bên phải là kết quả.

#### Xác định hình dạng vật thể:

Để tìm hình dạng của 1 vật thể trong ảnh, chúng tôi dùng thuật toán Canny. Thuật toán Canny cơ bản là một thuật toán phát hiện cạnh trong xử lý ảnh.



* Tôi chuyển hình ảnh sang ảnh xám đễ xử lý dễ dàng hơn.Dùng thuật toán Canny, tôi tìm được các cạnh của những vật thể.
* Tôi vẽ contour từ những cạnh tìm được. Sau đó, trong mỗi contour, tôi tính toán các cạnh của vật thể. ***cv2.arcLength()*** là một hàm trong thư viện OpenCV được sử dụng để tính toán **chu vi** của một đường viền (contour) được truyền vào. Tham số thứ hai của hàm (True) chỉ định rằng đường viền là đóng, tức là đường viền cuối cùng được nối với đường viền đầu tiên. Kết quả trả về là một giá trị thực đại diện cho chu vi của đường viền.Bằng cách nhân chu vi (epsilon) với 1% tôi tìm được **mức độ xấp xỉ của đa giác** đối với đường viền ban đầu. Lưu ý rằng Khi giá trị epsilon càng lớn, đa giác xấp xỉ sẽ càng đơn giản. Sau đó, tôi dùng hàm ***cv2.approxPolyDP()*** để xấp xỉ đường viền ban đầu (contour) thành một **đa giác** mới (approx), Kết quả của câu lệnh này là đa giác mới. Tiếp đến, tôi tính toán moments của đa giác. Hàm ***cv2.moments()*** dùng để tính toán các giá trị moment của đa giác xấp xỉ. Moments là một tập hợp các đặc trưng hình học của hình dạng, bao gồm diện tích, trọng tâm, và các moment bậc cao hơn. Hàm ***cv2.moments()*** trả về một từ điển chứa các giá trị moment. Từ các giá trị moment, tôi tìm đc tâm của mỗi vật thể.
* Từ tâm của vật thể tôi viết lên ảnh là hình dạng của nó.



Đây là ảnh chứa kết quả sau khi được xử lý.

## Báo cáo nhóm:

### Bài 1 – Giao tiếp máy tính nhúng Rasberry PI với vi điều khiển Arduino:

Thiết bị cần có:

* Dây truyền tín hiệu.
* Arduino Uno.
* Raspberry PI.
* Led, cảm biến nhiệt độ - độ ẩm.

#### Nguyên lý hoạt động:

Mục tiêu của bài này là điều khiển led thông qua arduino. Arduino sẽ nhận tín hiệu điều khiển từ Raspberry PI và trực tiếp điều khiển.

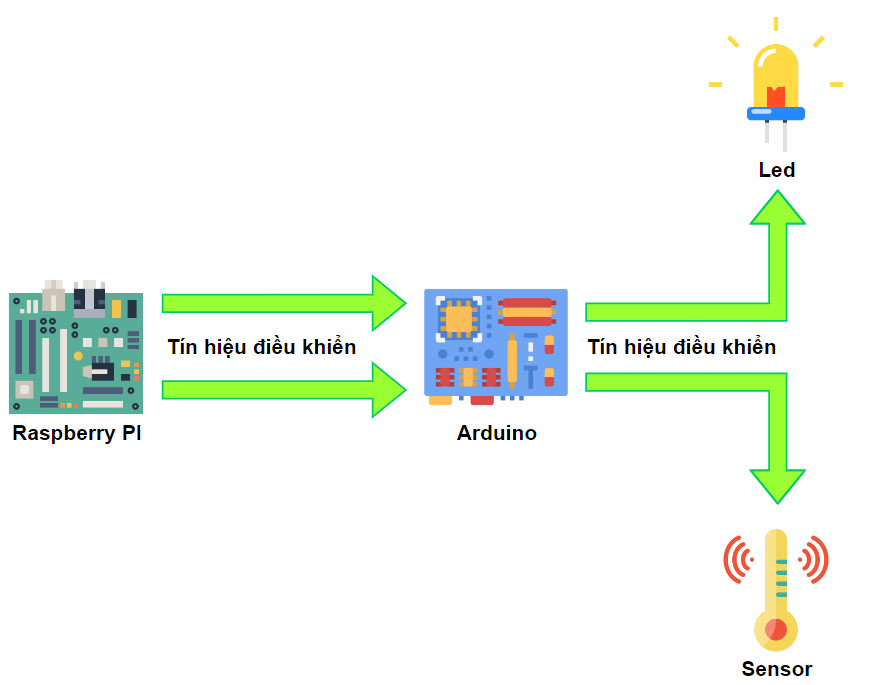
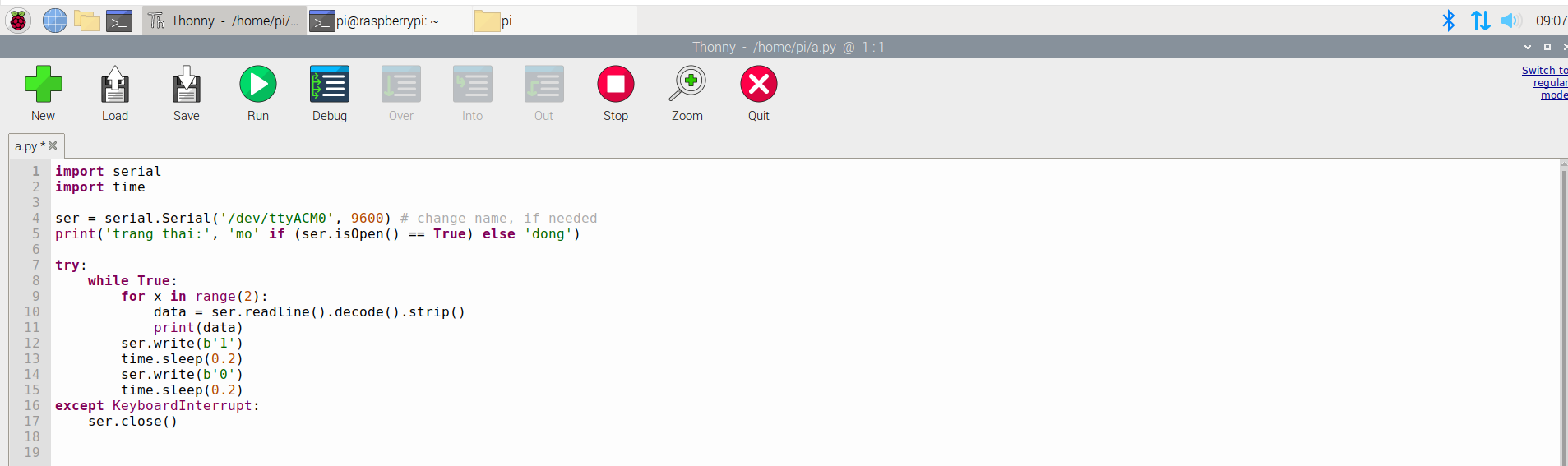


Figure 1- Sơ đồ nguyên lý bài 1

* Trên Rasberry PI: Được xem như 1 máy tính dùng để xử lý việc điều khiển đèn sáng/tắt.
* Trên Arduino: Dùng để tiếp nhận tín hiệu từ Raspberry PI và trực tiếp điều khiển led.

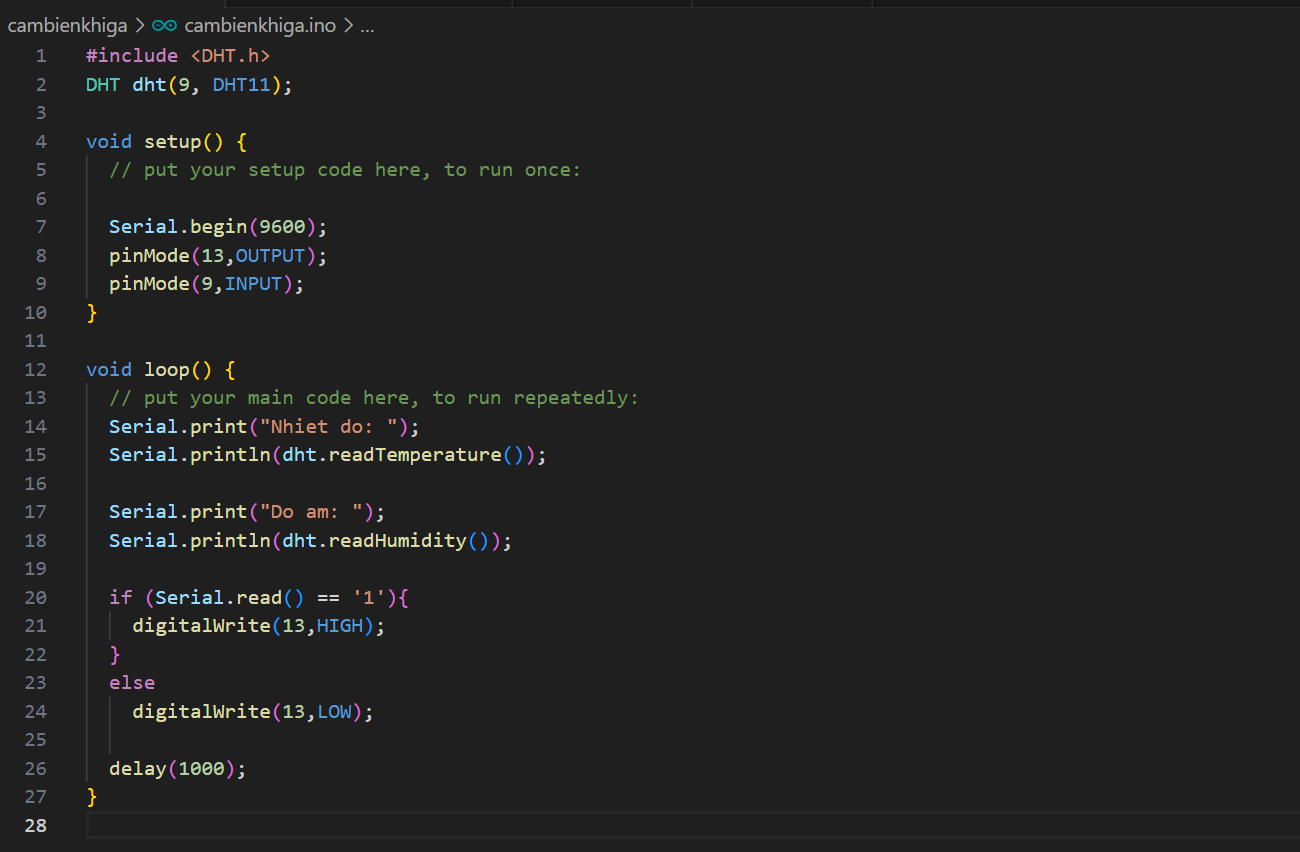
#### Chương trình phần mềm:

* Trên Raspberry PI:



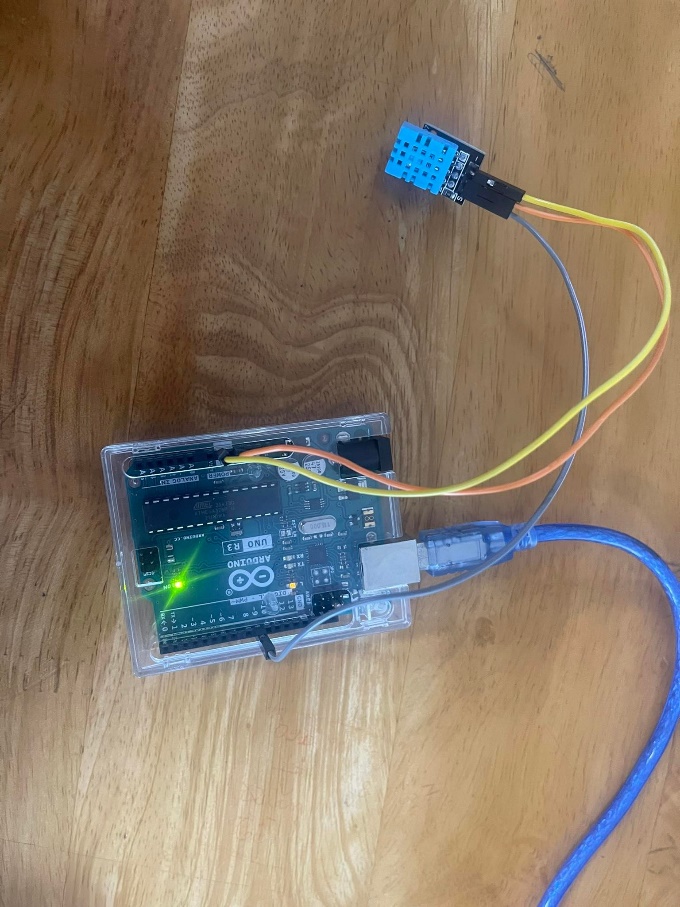
Tôi import 2 thư viện cần thiết là pyserial (thư viện hỗ trợ giao tiếp UART) và time (thư viện tích hợp các hàm liên quan đến date time và sleep). Tôi khởi tạo biến để kết nối Serial đến arduino. Sau đó trong vòng lặp True, tôi đọc dữ liệu từ arduino 2 lần, tiếp đến tôi gửi dữ liệu điều khiển con led (sáng tắt 1 lần).

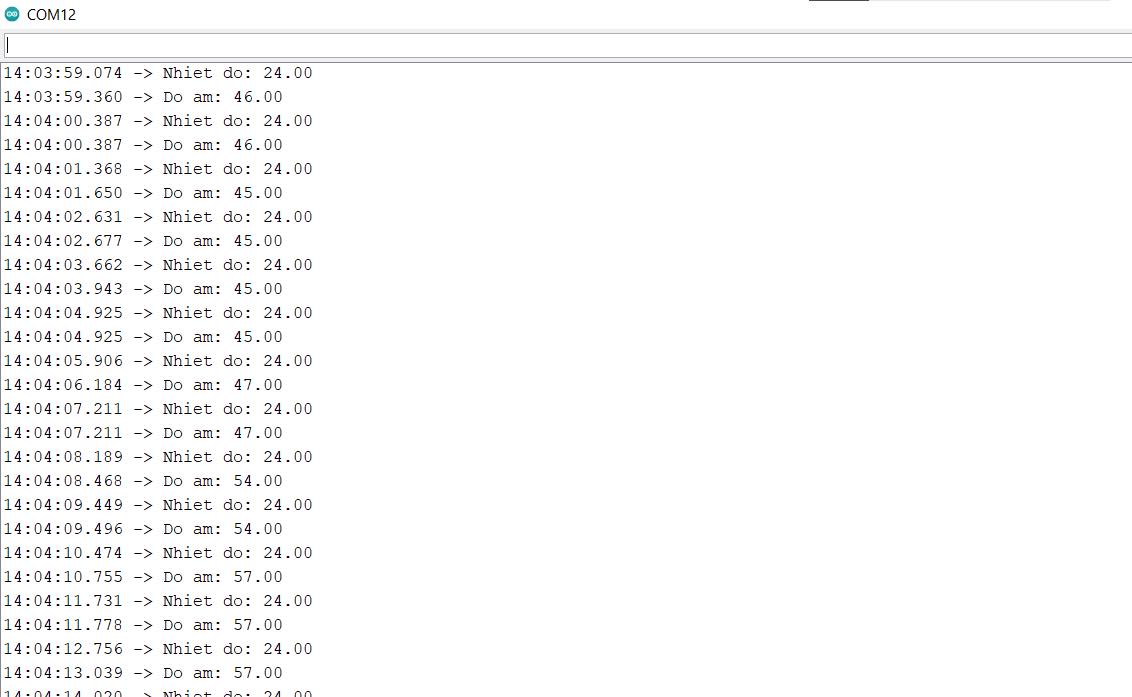
* Trên Arduino:



Tôi dùng thư viện DHT của cảm biến nhiệt độ - độ ẩm (thư viện tích hợp các hàm đọc dữ liệu từ cảm bién nhiệt độ - độ ẩm). Sau đó, trong vòng loop, tôi tiến hành đọc dữ liệu từ cảm biến và nhận tín hiệu từ Raspberry PI. Arduino sẽ trực tiếp điều khiển led sáng/tắt.

#### Kết quả:





### Bài 2 – Thiết kế giao diện giao tiếp với vi điều khiển Arduino bằng ngôn ngữ Python:

Thiết bị cần có:

* Máy tính bàn hoặc laptop.
* Arduino.
* Dây truyền tín hiệu.

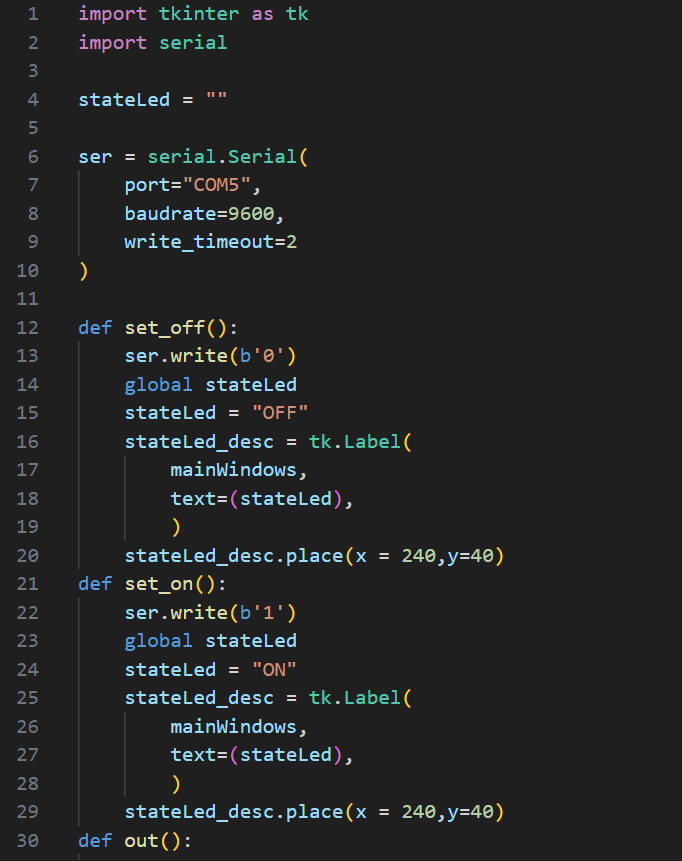
#### Nguyên lý hoạt động:

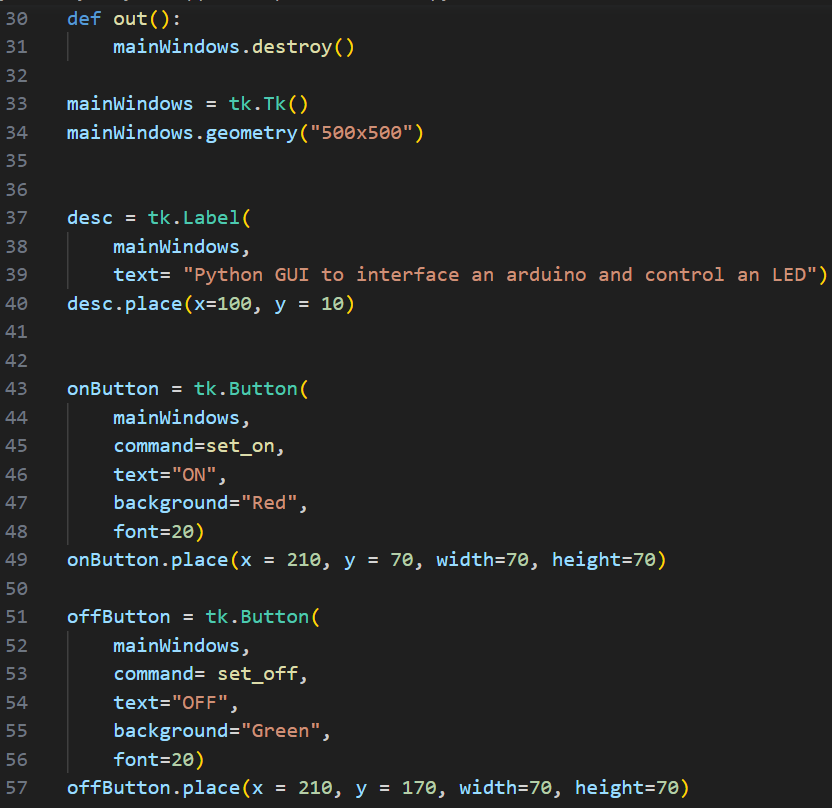
Mục đích của bài này là điều khiển led thông qua giao diện phần mềm trên máy được viết bằng python.

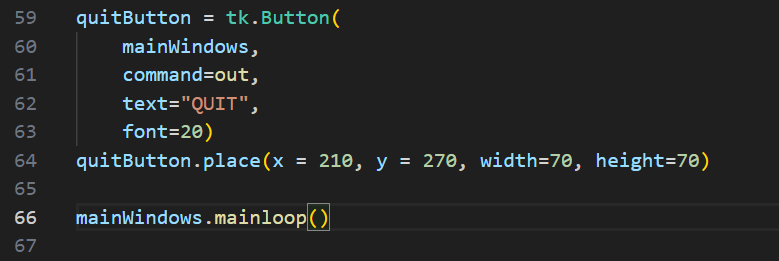


* Trên laptop: Xây dựng giao diện điều khiển led truyền tín hiệu tới arduino.
* Trên arduino: Thực hiện nhận tín hiệu và điều khiển trực tiếp led.

#### Chương trình phần mềm:

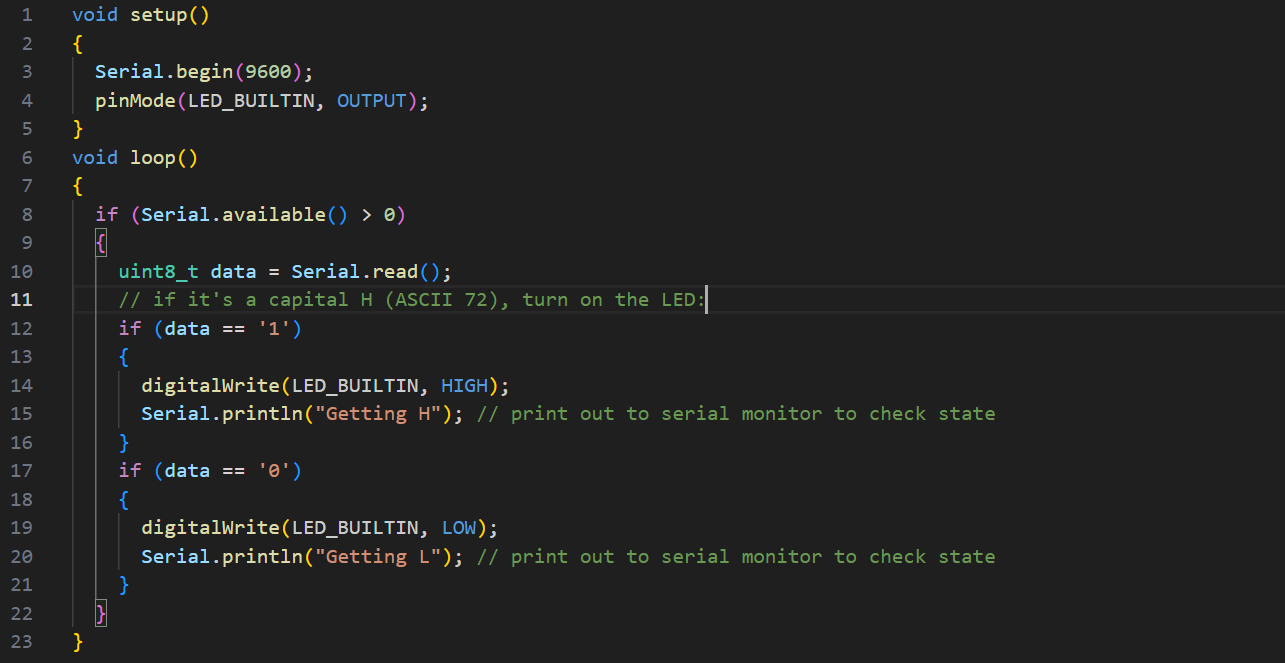
* Trên máy tính:





Tôi dùng thư viện tkinder (thư viện chuyên thiết kế phần mềm) và pyserial (thư viện hỗ trợ giao tiếp UART). Tôi khởi tạo biến để kết nối đến arduino. Sau đó, tôi tạo các thành phần của giao diện như button, label, liên kết sự kiện ấn nút vào các button. Sau khi thiết lập , tôi gọi hàm mainloop() để chạy giao diện.

* Trên Arduino:



Arduino sẽ thực hiện đọc dữ liệu từ máy tính và điều khiển led theo lệnh.

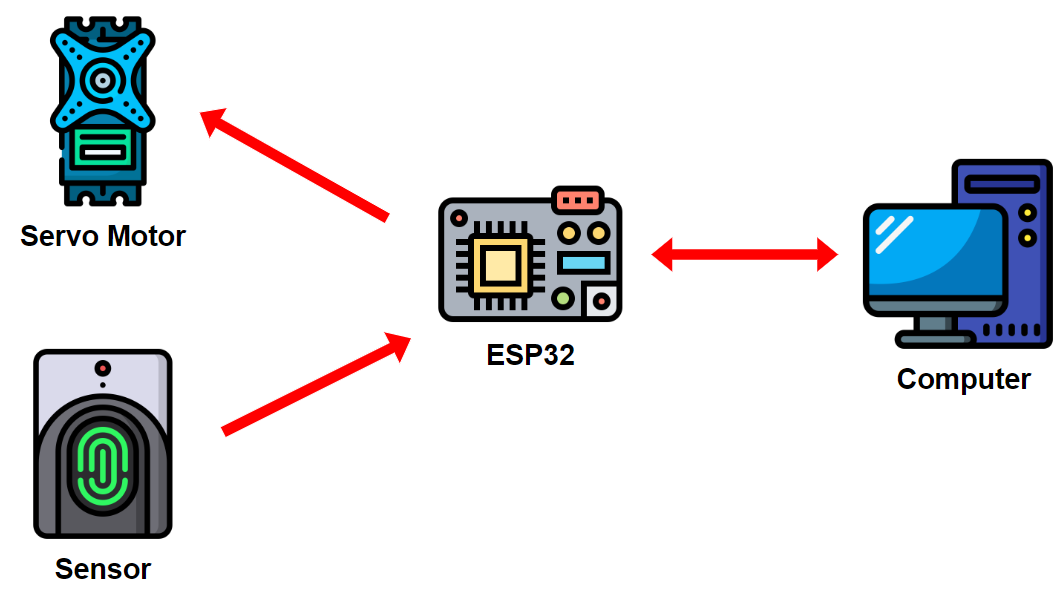
#### Kết quả:

### Đồ án cuối môn học –Hệ thống mở cửa bằng vân tay:

#### Ý tưởng – Mục tiêu:

* **Ý tưởng:** Ngày nay, sự chuyển dịch về công nghệ, phương pháp, cách thức hoạt động được diễn ra nhanh chóng và càng ngày xuất hiện nhiều. Chúng ta luôn tìm tòi, phát triển những thứ mới mẻ, luôn thay đổi những điều cũ kĩ, truyền thống, lạc hậu thành những thứ mới mẻ, hiện đại để phù hợp hơn với những nhu cầu trong thời kỳ 4.0. Cuốn theo làn sóng đó, tôi đã lên ý tưởng tạo ra 1 hệ thống mở cửa bằng vân tay. Hệ thống này giúp thay đổi cách thức mở cửa truyền thống, làm rút ngắn đi thời gian mở cửa, an toàn hơn và cũng như giám sát việc mở chặt chẽ hơn.
* **Mục tiêu:** 
  + Mở cửa bằng vân tay.
  + Giám sát hoạt động mở cửa qua ứng dụng được viết bằng ngôn ngữ lập trình python.

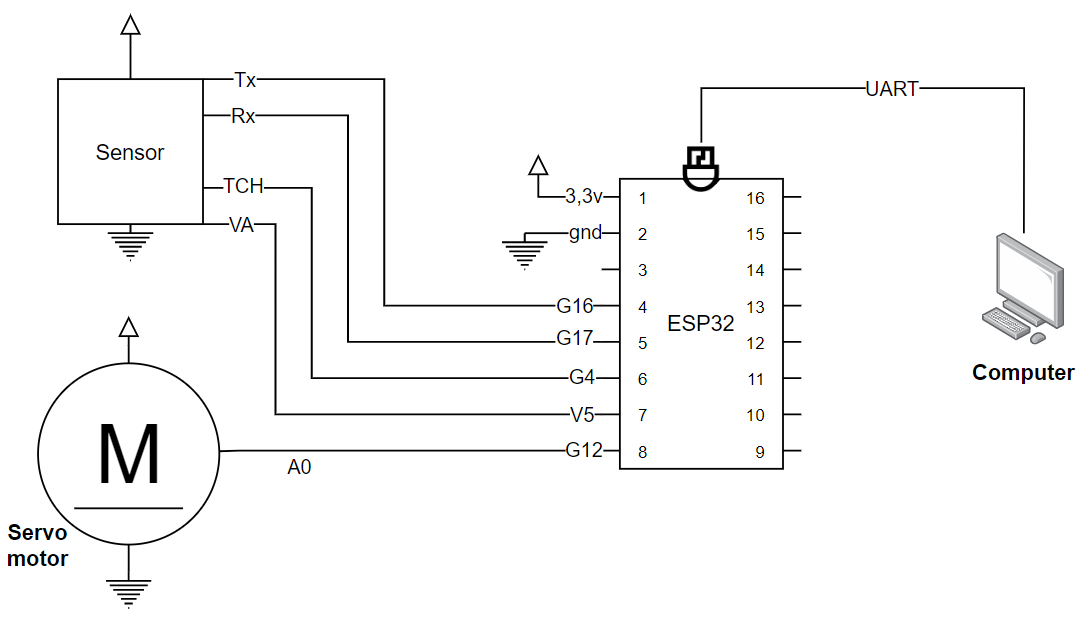
#### Nguyên lý hoạt động:



Để tổng hợp ý tưởng cho hệ thống quản lý cửa, chúng tôi đã mô phỏng nó bằng hệ thống như sau:

* **Máy tính:** là nơi lưu trữ mã nguồn phát triển ứng dụng quản lý cửa thông minh. Đây là nới hiển thị giao diện người dùng. Máy tính sẽ giao tiếp với vi điều khiển thông qua giao thức UART.
* **Vi điều khiển ESP32:** Để tăng tính ổn định cũng như khả năng mở rộng và phát triển, chúng tôi đã chọn vi điều khiển ESP32. Đây là một module phát triển ứng dụng IoT (Internet of Things) dựa trên vi xử lý ESPRESSIF. Nó cung cấp một nền tảng mạnh mẽ để phát triển các ứng dụng IoT thông qua việc kết nối với Wi-Fi, Bluetooth và các giao diện khác. Vi điều khiển này đóng vai trò là cầu nối giữa các thiết bị ngoại vi (cảm biến vân tay, động cơ) và máy tính. Nó sẽ giao tiếp máy tính qua giao tiếp UART.
* **Servo Motor:** Để mô phóng cho động cơ mở/đóng cửa, chúng tôi đã sử dụng động cơ servo. Đây là loại động cơ có sai số thấp và có dộ chính xác cao, thích hợp hơn cho hệ thông quản lý cửa thông minh. Động cơ sẽ tiếp nhận tín hiệu từ vi điều khiển và trực tiếp điều khiển cánh cửa.
* **Cảm biến vân tay:** là thành phần không thể thiếu trong dự án này, chúng tôi chọn cảm biến vân tay AS608 để phát triển dự án này. Đây là cảm biến sử dụng nguyên lý quang học để quét và nhận dạng các đặc trưng của vân tay. Nó sử dụng một đèn LED để chiếu sáng lên vân tay và một máy ảnh để chụp hình ảnh vân tay. Sau đó, các thuật toán được áp dụng để so sánh và nhận dạng vân tay dựa trên các điểm đặc trưng. Tuy AS608 không phải là một cảm biến vân tay cao cấp nhất, nhưng nó vẫn cung cấp một giải pháp đơn giản và hiệu quả cho nhận dạng vân tay trong các ứng dụng nhỏ và trung bình. Cảm biến sẽ gửi dữ liệu về vi đều khiển để xử lý.

#### Sơ đồ kết nối phần cứng:



Theo như sơ đồ, chúng tôi đã dùng các chân sau của esp32 điều khiển vfa truyền nhận dữ liệu:

* Chân nguồn 3,3 volt và chân GND: cấp nguồn cho cảm biến và động cơ.
* Chân nguồn 5 volt (V5): cấp nguồn cho Touch sensor.
* G16, G17: 2 chân tín hiệu dùng để giao tiếp UART với cảm biến.
* G4: nhận tín hiệu từ touch sensor của cảm biến.
* G12: truyền tín hiều điều khiển động cơ.
* USB: dùng để cấp nguồn cho vi điều khiển và giao tiếp với máy tính.

#### Chương trình phần mềm:

Toàn bộ mã nguồn cho dự án đều được chúng tôi lưu trên gitHub. Đây là mã QR quét đến kho lưu trữ mã nguồn của chúng tôi:



##### Bố cục folder:

Để thuận tiện cho việc quản lý dự án, chúng tôi đã phân chia các đoạn code thành những folder và file như sau:

##### Chương trình cho ESP32:

##### Chương trình ứng dụng:

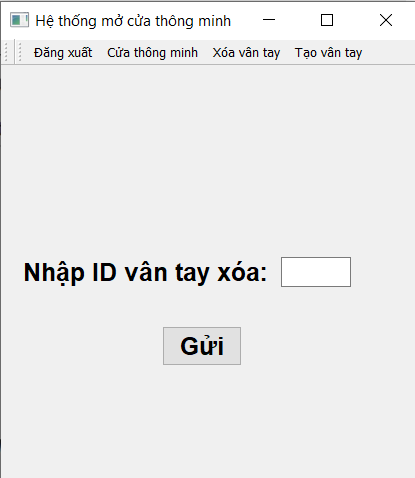
#### Kết quả - Thảo luận - Hướng phát triển:

**Kết quả:**









**Kết luận:**

* Ưu điểm:
  + **Tiện lợi và nhanh chóng:** Việc mở cửa trở nên tiện lợi và nhanh chóng hơn. Người dùng chỉ cần đặt ngón tay lên cảm biến và cửa sẽ mở mà không cần phải sử dụng chìa khóa hoặc mã PIN. Điều này rất thuận tiện, đặc biệt trong các khu vực có lưu lượng người ra vào lớn.
  + **Tăng cường bảo mật:** Cảm biến vân tay được coi là một trong những phương pháp xác thực an toàn nhất. Khi kết hợp với phần mềm quản lý cửa tự động, việc sử dụng cảm biến vân tay giúp đảm bảo rằng chỉ những người đã được đăng ký và có dấu vân tay được ủy quyền mới có thể mở cửa.
  + **Quản lý dễ dàng:** Cung cấp khả năng quản lý linh hoạt. Bạn có thể dễ dàng thấy ai là người sử dụng cửa từ hệ thống và có thể thêm hoặc xóa vân tay. Điều này giúp quản lý cửa một cách hiệu quả và thuận tiện.
* Nhược điểm:
  + **Giới hạn về độ chính xác:** Có thể xảy ra trường hợp cảm biến không nhận diện chính xác dấu vân tay của một số người dùng. Điều này có thể xảy ra do vấn đề về chất lượng của cảm biến hoặc tình trạng vân tay của người dùng. Khi đó, việc mở cửa có thể gặp khó khăn hoặc cần sự can thiệp của người quản lý.
  + **Sự cố kỹ thuật:** Một phần mềm quản lý cửa tự động phức tạp và nhạy cảm có thể gặp sự cố kỹ thuật. Nếu hệ thống gặp lỗi, có thể dẫn đến tình trạng cửa không hoạt động, mất kết nối hoặc lỗi trong việc xác thực người dùng. Điều này có thể gây khó khăn và phiền toái cho người sử dụng và yêu cầu sự can thiệp kỹ thuật để khắc phục.
  + **Chí phí và phụ thuộc vào công nghệ:** Triển khai và duy trì một hệ thống quản lý cửa tự động bằng phần mềm đòi hỏi đầu tư về cả phần cứng và phần mềm. Đồng thời, công nghệ liên quan có thể trở nên lạc hậu sau một thời gian ngắn, đòi hỏi việc cập nhật và nâng cấp để duy trì tính tương thích và an toàn.

**Hướng phát triển:**

* **Tích hợp công nghệ mới:** Hệ thống có thể tích hợp các công nghệ mới như nhận dạng khuôn mặt hoặc công nghệ nhận dạng sinh trắc học khác. Việc sử dụng nhiều phương pháp xác thực khác nhau tăng cường tính bảo mật và đa dạng hóa lựa chọn cho người dùng. Hệ thống sẽ kết nối với Internet of Things (IoT) để tương tác với các thiết bị thông minh khác trong mạng. Bằng cách sử dụng trí tuệ nhân tạo (AI), hệ thống có thể tự động học và thích ứng với các mô hình hành vi và ưu tiên của người dùng, cung cấp trải nghiệm cá nhân hóa và tối ưu hóa hiệu quả.
* **Kết hợp các hệ thống thông minh:** Hệ thống có thể được tích hợp với các hệ thống thông minh khác trong tòa nhà hoặc khu vực, chẳng hạn như hệ thống quản lý tòa nhà, hệ thống điều khiển ánh sáng, hệ thống an ninh, hoặc hệ thống điều khiển nhiệt độ. Điều này tạo ra một môi trường thông minh và tối ưu hóa hiệu suất năng lượng, cung cấp trải nghiệm tốt hơn cho người sử dụng và giảm chi phí hoạt động.
* **Quản lý từ xa và tích hợp di động**: Hệ thống có thể phát triển ứng dụng di động tương thích với nhiều nền tảng để người dùng có thể quản lý và kiểm soát cửa từ xa. Việc tích hợp các tính năng quản lý thông qua ứng dụng di động mang lại tính tiện lợi và linh hoạt cho người sử dụng.
* **Phân tích và thông báo thông minh:** Hệ thống có thể sử dụng phân tích dữ liệu để cung cấp thông tin chi tiết về lịch sử truy cập, thống kê sử dụng cửa, và các thông báo thông minh. Việc phân tích dữ liệu giúp người quản lý hiểu rõ hơn về hoạt động của hệ thống và đưa ra quyết định thông minh để cải thiện hiệu suất và an ninh.

# TÀI LIỆU THAM KHẢO:

1. Industrial Servo Control SystemsFundamentals And Applications, Revised And Expanded By [George W. Younkin](https://www.routledge.com/search?author=George%20W.%20Younkin)
2. AS608 Processor Datasheet Version 1.00, September 2015 By Hangzhou Synochip Data Security Technology Co., Ltd.
3. ESP32 Datasheet (PDF) - ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD.
4. Qt5 Python Tutorial (<https://www.tutorialspoint.com/pyqt5/pyqt5_quick_guide.html>)
5. OpenCV Python Tutorial

(

<https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html)>

1. Building Internet of Things with the By Charalampos Doukas, Manolis Tsaramirsis